

CLAIMS

What is claimed is:

- 1 1. A method of generating code to be deployed in an application server,
- 2 comprising the steps of:
- 3 receiving an archive file to be deployed;
- 4 introspecting an input class included in the archive file to generate
- 5 information relating to the input class;
- 6 generating a markup language description of the input class based on the
- 7 generated information relating to the input class;
- 8 creating an event handler for method nodes found in the markup language
- 9 description;
- 10 registering the event handler;
- 11 parsing the markup language description and invoking the registered
- 12 event handler; and
- 13 generating output code using the invoked event handler.

- 1 2. The method of claim 1, wherein the class file is an Enterprise Java Bean
- 2 class file.

1 3. The method of claim 2, wherein the step of introspecting an input class
2 included in the archive file comprises the steps of:

3 extracting information identifying methods included in the input class;

4 and

5 for each method, extracting information relating to parameters of the
6 method.

1 4. The method of claim 3, wherein the step of generating a markup language
2 description of the input class comprises the step of:

3 generating an Extensible Markup Language description of the input class
4 based on the generated information relating to the input class.

1 5. The method of claim 4, wherein the step of creating an event handler
2 comprises the step of:

3 creating a Simple Application Programming Interface for Extensible
4 Markup Language event handler for a method node found in the markup
5 language description

1 6. The method of claim 5, wherein the step of parsing the markup language
2 description and invoking the registered event handler comprises the step of:
3 parsing the markup language description using a Simple Application
4 Programming Interface for Extensible Markup Language parser and invoking the
5 registered Simple Application Programming Interface for Extensible Markup
6 Language event handler.

1 7. The method of claim 1, wherein the step of creating an event handler
2 comprises the step of:
3 creating a plurality of event handlers for a method node found in the
4 markup language description.

1 8. The method of claim 7, wherein the step of registering the event handler
2 comprises the step of:
3 registering each of the plurality of event handlers.

1 9. The method of claim 8, wherein the step of parsing the markup language
2 description and invoking the registered event handler comprises the step of:

3 parsing the markup language description and invoking each of the
4 plurality of registered event handlers.

1 10. The method of claim 9, wherein the step of generating output code
2 comprises the step of:

3 generating output code using each of the plurality of invoked event
4 handler in parallel.

1 11. The method of claim 10, wherein the archive file is an Enterprise Java
2 Bean archive file.

1 12. The method of claim 11, wherein the step of introspecting an input class
2 included in the archive file comprises the steps of:

3 extracting information identifying methods included in the input class;
4 and

5 for each method, extracting information relating to parameters of the
6 method.

1 13. The method of claim 12, wherein the step of generating a markup
2 language description of the input class comprises the step of:

3 generating an Extensible Markup Language description of the input class
4 based on the generated information relating to the input class.

1 14. The method of claim 13, wherein the step of creating a plurality of event
2 handlers comprises the step of:

3 creating a plurality of Simple Application Programming Interface for
4 Extensible Markup Language event handlers for a method node found in the
5 markup language description

1 15. The method of claim 14, wherein the step of parsing the markup language
2 description and invoking the registered event handler comprises the step of:

3 parsing the markup language description using a Simple Application
4 Programming Interface for Extensible Markup Language parser and invoking the
5 plurality of registered Simple Application Programming Interface for Extensible
6 Markup Language event handlers.

1 16. A system for generating code to be deployed in an application server
2 comprising:
3 a processor operable to execute computer program instructions;
4 a memory operable to store computer program instructions executable
5 by the processor; and
6 computer program instructions stored in the memory and executable to
7 perform the steps of:
8 receiving an archive file to be deployed;
9 introspecting an input class included in the archive file to generate
10 information relating to the input class;
11 generating a markup language description of the input class based on the
12 generated information relating to the input class;
13 creating an event handler for a method node found in the markup
14 language description;
15 registering the event handler;
16 parsing the markup language description and invoking the registered
17 event handler; and
18 generating output code using the invoked event handler.

1 17. The system of claim 16, wherein the archive file is an Enterprise Java
2 Bean archive file.

1 18. The system of claim 17, wherein the step of introspecting an input class
2 included in the archive file comprises the steps of:

3 extracting information identifying systems included in the input class; and
4 for each system, extracting information relating to parameters of the
5 system.

1 19. The system of claim 18, wherein the step of generating a markup language
2 description of the input class comprises the step of:

3 generating an Extensible Markup Language description of the input class
4 based on the generated information relating to the input class.

1 20. The system of claim 19, wherein the step of creating an event handler
2 comprises the step of:

3 creating a Simple Application Programming Interface for Extensible
4 Markup Language event handler for a system node found in the markup language
5 description

1 21. The system of claim 20, wherein the step of parsing the markup language
2 description and invoking the registered event handler comprises the step of:
3 parsing the markup language description using a Simple Application
4 Programming Interface for Extensible Markup Language parser and invoking the
5 registered Simple Application Programming Interface for Extensible Markup
6 Language event handler.

1 22. The system of claim 21, wherein the step of creating an event handler
2 comprises the step of:
3 creating a plurality of event handlers for a system node found in the
4 markup language description.

1 23. The system of claim 22, wherein the step of registering the event handler
2 comprises the step of:
3 registering each of the plurality of event handlers.

1 24. The system of claim 23, wherein the step of parsing the markup language
2 description and invoking the registered event handler comprises the step of:

3 parsing the markup language description and invoking each of the
4 plurality of registered event handlers.

1 25. The system of claim 24, wherein the step of generating output code
2 comprises the step of:

3 generating output code using each of the plurality of invoked event
4 handler in parallel.

1 26. The system of claim 25, wherein the archive file is an Enterprise Java
2 Bean archive file.

1 27. The system of claim 26, wherein the step of introspecting an input class
2 included in the archive file comprises the steps of:
3 extracting information identifying systems included in the input class; and
4 for each system, extracting information relating to parameters of the
5 system.

1 28. The system of claim 27, wherein the step of generating a markup language
2 description of the input class comprises the step of:

3 generating an Extensible Markup Language description of the input class
4 based on the generated information relating to the input class.

1 29. The system of claim 28, wherein the step of creating a plurality of event
2 handlers comprises the step of:

3 creating a plurality of Simple Application Programming Interface for
4 Extensible Markup Language event handlers for a system node found in the
5 markup language description

1 30. The system of claim 29, wherein the step of parsing the markup language
2 description and invoking the registered event handler comprises the step of:

3 parsing the markup language description using a Simple Application
4 Programming Interface for Extensible Markup Language parser and invoking the
5 plurality of registered Simple Application Programming Interface for Extensible
6 Markup Language event handlers.

1 31. A computer program product for generating code to be deployed in an
2 application server comprising:

3 a computer readable medium;

4 computer program instructions, recorded on the computer readable
5 medium, executable by a processor, for performing the steps of
6 receiving an archive file to be deployed;
7 introspecting an input class included in the archive file to generate
8 information relating to the input class;
9 generating a markup language description of the input class based on the
10 generated information relating to the input class;
11 creating an event handler for a method node found in the markup
12 language description;
13 registering the event handler;
14 parsing the markup language description and invoking the registered
15 event handler; and
16 generating output code using the invoked event handler.

1 32. The computer program product of claim 31, wherein the archive file is an
2 Enterprise Java Bean archive file.

1 33. The computer program product of claim 32, wherein the step of
2 introspecting an input class included in the archive file comprises the steps of:

3 extracting information identifying computer program products included in
4 the input class; and
5 for each computer program product, extracting information relating to
6 parameters of the computer program product.

1 34. The computer program product of claim 33, wherein the step of
2 generating a markup language description of the input class comprises the step
3 of:
4 generating an Extensible Markup Language description of the input class
5 based on the generated information relating to the input class.

1 35. The computer program product of claim 34, wherein the step of creating
2 an event handler comprises the step of:
3 creating a Simple Application Programming Interface for Extensible
4 Markup Language event handler for a computer program product node found in
5 the markup language description

1 36. The computer program product of claim 35, wherein the step of parsing
2 the markup language description and invoking the registered event handler
3 comprises the step of:

4 parsing the markup language description using a Simple Application
5 Programming Interface for Extensible Markup Language parser and invoking the
6 registered Simple Application Programming Interface for Extensible Markup
7 Language event handler.

1 37. The computer program product of claim 31, wherein the step of creating
2 an event handler comprises the step of:
3 creating a plurality of event handlers for a computer program product
4 node found in the markup language description.

1 38. The computer program product of claim 37, wherein the step of
2 registering the event handler comprises the step of:
3 registering each of the plurality of event handlers.

1 39. The computer program product of claim 38, wherein the step of parsing
2 the markup language description and invoking the registered event handler
3 comprises the step of:

4 parsing the markup language description and invoking each of the
5 plurality of registered event handlers.

1 40. The computer program product of claim 39, wherein the step of
2 generating output code comprises the step of:

3 generating output code using each of the plurality of invoked event
4 handler in parallel.

1 41. The computer program product of claim 40, wherein the archive file is an
2 Enterprise Java Bean archive file.

1 42. The computer program product of claim 41, wherein the step of
2 introspecting an input class included in the archive file comprises the steps of:
3 extracting information identifying computer program products included in
4 the input class; and

5 for each computer program product, extracting information relating to
6 parameters of the computer program product.

1 43. The computer program product of claim 42, wherein the step of
2 generating a markup language description of the input class comprises the step
3 of:

4 generating an Extensible Markup Language description of the input class
5 based on the generated information relating to the input class.

1 44. The computer program product of claim 43, wherein the step of creating a
2 plurality of event handlers comprises the step of:

3 creating a plurality of Simple Application Programming Interface for
4 Extensible Markup Language event handlers for a computer program product
5 node found in the markup language description

1 45. The computer program product of claim 44, wherein the step of parsing
2 the markup language description and invoking the registered event handler
3 comprises the step of:

4 parsing the markup language description using a Simple Application
5 Programming Interface for Extensible Markup Language parser and invoking the
6 plurality of registered Simple Application Programming Interface for Extensible
7 Markup Language event handlers.